Memetic Differential Evolution for Constrained Numerical Optimization Problems

Saúl Domínguez-Isidro
National Laboratory on
Advanced Informatics (LANIA) A.C.
Xalapa, Veracruz, México
Email: sdominguez@lania.mx

Efrén Mezura-Montes
Departamento de Inteligencia Artificial
Universidad Veracruzana
Xalapa, Veracruz, México
Email: emezura@uv.mx

Guillermo Leguizamón
Departamento de Informática
Universidad Nacional de San Luis,
San Luis, Argentina
Email: legui@unsl.edu.ar

***Abstract*—This paper presents a memetic algorithm for solving constrained numerical optimization problems. The proposed approach uses differential evolution as a global search algorithm, which was improved with a mathematical programming method called Powell's conjugate direction as a local search operator. To the best of the authors' knowledge, this is the first attempt to use such mathematical programming method within differential evolution for constrained optimization. The proposed algorithm was tested on 36 test problems used in the special session on "Single Objective Constrained Real-Parameter Optimization" in CEC'2010. The proposed algorithm is able to find competitive results with respect to the winner algorithm in that session.**

I. Introduction

Constrained numerical optimization problems (CNOPs), also known as constrained nonlinear optimization problems, are usually found in complex real-world instances such as in engineering design, protein folding in bioinformatics, logistics process, etc. [1]. Hence, the use of heuristic-based search algorithms to solve CNOPs is very common nowadays. There is a considerable amount of research regarding evolutionary algorithms (EAs) [2] and swarm intelligence algorithms (SIAs) [3] to solve CNOPs as it is outlined in a recent review on the topic [4]. In such review, the importance of local search operators was remarked. However, there are classic optimization methods which have not been used for local search purposes within an EA to solve CNOPs. This is the precisely the motivation of this work, where the Powell's conjugate direction method (CMD) [5], which, to the best of the authors' knowledge has not been adopted within an EA to solve CNOPs, is coupled with differential evolution [6] so as to generate a competitive memetic algorithm to deal with constrained numerical search spaces. There is one approach where CMD was adopted as a local search operator [7], but it was applied for unconstrained nonlinear optimization problems.

A memetic algorithm (MA) is based on the interaction of two processes, global and local search, in a particular algorithm to get the advantage of both searches, i.e., a MA is the suitable synergy of two different algorithms, in our case DE plus CDM. Nowadays, the usage of MAs to solve optimization problems has become very popular because their competitive performance in complex search spaces [8].

The design and addition of a suitable constraint-handling mechanism within an EA or SIA when solving a CNOP is an open problem [9]. A recently proposed taxonomy of constraint-handling techniques group them in two generations [4], where those first-generation techniques were centered on penalty functions [10], decoders [11], special operators [12], and separation of objectives and constraints [13].

Based on the shortcomings of those first-generation constraint-handling techniques (e.g. careful fine-tuning of additional parameters, difficult implementations, high computational cost, premature convergence) a second generation of techniques has been proposed, where the most representative approaches are the feasibility rules [14], the stochastic ranking [15] and the ε-constrained method [16]. This last technique was adopted in this work because it is the same used in the best approach presented in the CEC'2010 competition on constrained-real parameter optimization. The *ε-constrained method* transforms the CNOP into an unconstrained numerical optimization problem by using a so-called ε level comparison, which will be detailed later in this document.

The contents of this paper are organized as follows: the problem statement is described in Section II. In Section III the DE algorithm is introduced. Section IV presents a brief review of DE-based memetic algorithms for CNOPs, while in Section V the memetic DE proposed in this work is detailed. The experiments and results are summarized in Section VI. Finally, in Section VII, some conclusions are shown and the future work is defined.

II. Problem Statement

The problem of interest in this paper is the constrained numerical optimization problem (CNOP), which, without loss of generality, can be defined as to:

$$\text{Minimize: } f(\vec{x}) \tag{1}$$

Subject to

$$g_i(\vec{x}) \leq 0, \qquad i = 1, \ldots, m \tag{2}$$

$$h_j(\vec{x}) = 0, \qquad j = 1, \ldots, p \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^n$ is the vector of solutions, m is the number of inequality constraints, and p is the number of equality constraints. The search space $\mathcal{S}$ is defined by lower and upper limits $L_k \leq x_k \leq U_k$ for each $x_k \in \vec{x}$. The feasible region $\mathcal{F}$ of the search space $\mathcal{S}$ is the set of all solutions which satisfy all the constraints (Eqs. 2 and 3). Equality constraints are usually transformed into inequality constraints as follows

[17]: $|h_j(\vec{x})| - \delta \leq 0$, where δ is the tolerance allowed (a very small value).

III. Differential Evolution Algorithm (DE)

Differential Evolution was proposed by Storn and Price [6] as a highly competitive EA designed to solve unconstrained global optimization problems over continuous search spaces (see Algorithm 1). In DE, a solution to the optimization problem is known as a vector, detailed in Eq. (4):

$$X_{G,i} = (x_{G,i,j}, \ldots, x_{G,i,D}) \tag{4}$$

where $X_{G,i}$ represents a vector i at generation G, and D is the number of decision variables (i.e. the search space dimensionality, $D = n$). In the same way, a population is represented as shown in Eq. (5):

$$P_G = (X_{G,i}, \ldots, X_{G,Pmax}) \tag{5}$$

where $Pmax$ is the fixed population size at each generation G. $Gmax$ is the maximum number of generations (i.e. cycles) of the algorithm.

For each vector i in the population (known as target vector $X_{G,i}$), an offspring (trial vector) is generated by using a mutation and a crossover operators. The mutation operator works as follows: (1) three vectors $X_{G,r0}$, $X_{G,r1}$ and $X_{G,r2}$, where $r0 \neq r1 \neq r2 \neq i$, are chosen at random from the current population; after that, (2) the difference vector of $X_{G,r1}$ and $X_{G,r2}$ is computed and then scaled by a user-defined factor $F > 0$; finally, (3) the scaled difference vector is added to the third vector $X_{G,r0}$. As a result, a mutation vector: $V_{G,i} = X_{G,r0} + F(X_{G,r1} - X_{G,r2})$ is computed. The aim of the mutation operator is generating new vectors based on search directions defined by the position of vectors in the current population.

After the mutation vector was obtained, a trial vector $U_{G,i}$ is generated by a crossover operator using the target vector $X_{G,i}$ and the mutation vector $V_{G,i}$ as indicated in Eq. (6):

$$u_{G,i,j} = \begin{cases} v_{G,i,j} & \text{if}(rand_i \leq Cr) \text{ or } (j = J_{rand}) \\ x_{G,i,j} & \text{otherwise} \end{cases} \tag{6}$$

where $Cr \in [0, 1]$ is the crossover parameter, which controls the number of decision variables to be copied into the trial vector from the mutation vector or from the target vector. $J_{rand} \in [1 - D]$ is an integer random number which ensures that at least one element of the mutation vector is copied to the trial vector so as to prevent the trial vector to be a copy of the target vector.

Finally the replacement process is given in Eq. (7), where the best vector between target and trial remains for the next generation of the algorithm:

$$X_{G+1,i} = \begin{cases} U_{G,i} & \text{if}(f(U_{G,i}) \leq f(X_{G,i})), \\ X_{G,i} & \text{otherwise} \end{cases} \tag{7}$$

The complete steps of DE are presented in Algorithm 1.

Algorithm 1 Differential Evolution Algorithm (DE/rand/1/bin)

```
Randomly generate an initial population of vectors P_0 = (X_{0,i}, ..., X_{0,Pmax})
Calculate the fitness of each vector in the initial population.
repeat
    for i ← 1, Pmax do
        Randomly select r0,r1,r2 ∈ [1, Pmax] and r0 ≠ r1 ≠ r2 ≠ i
        Randomly select J_rand ∈ [1, D]
        for j ← 1, D do
            if rand_j ≤ Cr Or j = J_rand then
                u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} − x_{G,r2,j})
            else
                u_{G,i,j} = x_{G,i,j}
            end if
        end for
        if f(U_{G,i}) ≤ f(X_{G,i}) then
            X_{G+1,i} = U_{G,i}
        else
            X_{G+1,i} = X_{G,i}
        end if
    end for
    G = G + 1
until a stop condition is satisfied
```

IV. Related Work

Nowadays, it is a common practice the usage of local search in nature-inspired algorithms (NIAs), i.e. both, EAs and SIAs, for solving different instances of CNOPs [4]. Moreover, in a taxonomy of problems tackled by MAs [8] the CNOPs are indeed included. In this section we will briefly mention those studies where DE has been coupled with different local search operators to sample constrained continuous search spaces.

Liu et al. in [18] proposed a co-evolution-memetic-based algorithm. They used two different population within DE. One population aimed to minimize the objective function, regardless of constraints, and the other one looked for minimizing the violation of constraints, regardless of the objective function. Gaussian mutation was also applied to the individuals in the populations as a local-search-like operator. Takahama and Sakai in [19] used DE with exponential crossover (known as DE/rand/1/exp) and the ε-constrained method to solve CNOPs. They also adopted a gradient-based mutation to their approach as a local search operator. The authors presented an improved version based on a new control mechanism for the ε-tolerance [20]. In such recent algorithm, they also proposed two novel mechanisms to control boundary constraints to further improve the performance of their approach. Mandal and Nagaratnam in [21] used an hybrid-mutation strategy. The authors combined two techniques in the mutation process, which were: DE/current-to-best/2 and DE/rand/1/bin. Moreover, the authors added the Solis and Wets algorithm proposed in [22] as a local search operator. Fuqing Zhao et al. [23] proposed a memetic DE. In this case, a mutation operator based on the Cauchy distribution was used as a local search operator. They also implemented a self-adaptive value for the scaling factor used in the differential mutation operator. Pescador and Coello Coello [24] implemented a crossover-memetic-based algorithm. They implemented DE as the global search algorithm and the simplex crossover as the local search operator. The authors applied the local search operator in the neighborhood of the

best and also the worst solutions of the population. Pan et al. [25] proposed a DE algorithm with a Cauchy mutation applied to the best solution in the current population as a local search operator. The approach was designed to solve, besides CNOPs, other optimization problems such as unconstrained large-scale optimization problems. Hernández et al. [26] proposed a memetic algorithm using DE for global search and a hill-climbing method as local search operator. After the selection process in DE, the local search was applied to the best solution of the population. Muelas et al. [27] proposed a DE-based memetic algorithm with the so-called multiple trajectory search algorithm as the local search operator to solve numerical optimization problems. Menchaca and Coello Coello in [28] proposed a hybrid DE algorithm. They combined the Nelder-Mead method with DE to sample constrained search spaces. This approach is not precisely a MA because the Nelder-Mead method is used for global search purposes as DE.

V. Proposed Approach

As it was pointed out in Section IV, DE has been combined with different types of local search operators (e.g. mutation operators, crossover operators, direct-search methods, indirect-search methods). However, among those direct-search methods the Powell's conjugate direction method has not been considered. This section describes how such method is adopted as a local search operator within DE by using the ε-constrained method to deal with the feasibility information in a CNOP. The conjugate direction method is described firstly. Afterwards, the ε-constrained method is introduced as well as the equality constraint tolerance value handling. Finally, the integration of the conjugate direction method within DE is detailed.

A. Local search operator

The conjugate direction method (CDM) was proposed by Powell in 1964 [5]. CMD is a deterministic method to solve optimization problems without calculating derivatives. This method has a convergence proof for quadratic objective functions, but it also works for non-quadratic functions [29]. The aim of CDM is to generate a set of $\boldsymbol{N}$ linearly independent search directions to perform a series of unidirectional searches along each of these search directions, starting each time from the previous point. The unidirectional search method adopted in this work was the Golden search method. The pseudocode of the adapted CDM used in this work is detailed in Algorithm 2, where an additional stopping criterion, based on a iteration counter limited by the number of variables of the CNOP $(T > N+1)$ was added. The aim of the local search operator is to explore new solutions in the search space which increase the capabilities of DE.

B. Constraint-handling

The ε-constrained method was proposed by Takahama and Sakai [16]. This constraint-handling technique transforms a CNOP into an unconstrained optimization problem. The constraint violation ϕ of a given vector $X_{G,k}$ is computed as the sum of the amounts of all violated constraints (see Eq. 8).

$$\phi(X_{G,k}) = \sum_{i=1}^{m} max(0, g_i(X_{G,k})) + \sum_{j=1}^{p} max(0, |h_j(X_{G,k})| - \delta) \tag{8}$$

Algorithm 2 Updated Powell's Conjugate Direction Method used in this work.

```
Choose a starting point X^(0) and a set of N linearly independent directions S^(i) i = 1, 2, ..., N.
T = 0
repeat
    Set λ using the Golden Search Algorithm
    for i ← 1, N do
        Set X^i = X^(i-1) + λS^i
    end for
    Form a new conjugate direction d = X^(N) − X^(1)
    Replace S^(j) = S^(j-1) for all j = N, N − 1, ..., 2.
    Set S^(1) = d / ||d||.
until ||d|| ≤ α OR search directions are linearly dependent OR T > N + 1
```

The ε tolerance allows the so-called $\leq_\varepsilon$ comparison between two vectors by using only their fitness function values, e.g., $f(X_{G,k})$ and $f(X_{G,l})$, when the ϕ values of both solutions lie below the ε value, or when both ϕ values are equal (see Eq. 9).

$$X_{G,k} \leq_\varepsilon X_{G,l} \Leftrightarrow \begin{cases} f(X_{G,k}) \leq f(X_{G,l}) & \text{if } \phi(X_{G,k}), \phi(X_{G,l}) \leq \varepsilon \\ f(X_{G,k}) \leq f(X_{G,l}) & \text{if } \phi(X_{G,k}) = \phi(X_{G,l}) \\ \phi(X_{G,k}) \leq \phi(X_{G,l}) & \text{,otherwise} \end{cases} \tag{9}$$

The ε level is dynamically decreased at each generation as indicated in Eq. 10.

$$\varepsilon(G) = \begin{cases} \varepsilon(0)(1 - \frac{G}{Gc})^{cp} & , 0 < G < Gc \\ 0 & , G \geq Gc \end{cases} \tag{10}$$

where cp is a user-defined parameter to control the reduction speed of the ε tolerance, G is the current generation number and Gc is the generation number when the ε value is set to zero.

The initial ε value (i.e., $\varepsilon(0)$) is the constraint violation of the best solution in the initial population, i.e., $\phi(X_{0,\theta})$ as indicated in Eq. 11.

$$\varepsilon(0) = \phi(X_{0,\theta}) \tag{11}$$

C. Tolerance value for equality constraints

Recalling from Section II, equality constraints are often transformed into inequality constraints using a δ tolerance. Inspired in [30], the δ tolerance decreases at each iteration as indicated in Eq. 12 until a suitable value is reached.

$$\delta(G+1) = \frac{\delta(G)}{dec} \tag{12}$$

where dec is a user-defined parameter which determines how fast the δ value decreases over time and $\delta(0)$ is also a user-defined parameter.

D. Memetic differential evolution with Powell's conjugate direction method

The combination of DE with CDM, called MEMetic DE with Powell's method (MEMDEP for short) is shown in Algorithm 3. The local search is carried out by CDM as follows: after applying the differential mutation, crossover and DE's selection process, three vectors from the population are selected: the best, the worst, and another one chosen at random. The CMD local search is applied to each one of these vector separately. The vector obtained by the local search operator replaces the original vector in the three cases if it is better based on the ε-constrained method.

Algorithm 3 MEMDEP

Randomly generate an initial population of vectors $P_0 = (X_{0,i}, \ldots, X_{0,Pmax})$
Calculate the fitness of each vector in the initial population.
Set the ε value with Eq. 11
repeat
  for $i \leftarrow 1, Pmax$ **do**
    Randomly select $r0$,$r1$,$r2 \in [1, Pmax]$ and $r0 \neq r1 \neq r2 \neq i$
    Randomly select $J_{rand} \in [1, D]$
    for $j \leftarrow 1, D$ **do**
      if $rand_j \leq Cr$ Or $j = Jrand$ **then**
        $u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j})$
      else
        $u_{G,i,j} = x_{G,i,j}$
      end if
    end for
    if $U_{G,i} \leq_{\varepsilon} X_{G,i}$ (see Eq. 9) **then**
      $X_{G+1,i} = U_{G,i}$
    else
      $X_{G+1,i} = X_{G,i}$
    end if
  end for
  Select the best solution of the population $\mathbf{X_{best}}$
  Select the worst solution of the population $\mathbf{X_{worst}}$
  Randomly select $\mathbf{q} \in [\mathbf{1}, \mathbf{Pmax}]$ **and** $\mathbf{q} \neq \mathbf{X_{worst}} \neq \mathbf{X_{best}}$
  Apply local search to $\mathbf{X_{G,best}}$, $\mathbf{X_{G,worst}}$ **and** $\mathbf{X_{G,q}}$ **using Algorithm 2**.
  Compare each vector $(\mathbf{best}, \mathbf{worst}, \mathbf{q})$ **against the vectors generated by Algorithm 2** $(\mathbf{best}', \mathbf{worst}', \mathbf{q}')$ **using Eq. 9 and keep the best ones**.
  if There are equality constraints **then**
    decrease δ using Eq. 12
  end if
  Update ε value with Eq. 10
  $G = G + 1$
until *Max_FES* is reached

VI. Experiments and results

The algorithm was tested in thirty six benchmark problems used in the special session on "Single Objective Constrained Real-Parameter Optimization" in CEC'2010 [31]. 25 independent runs were performed for each test problem. The experiments consist on two phases: (1) analyzing the algorithm's behavior by computing statistical values and (2) comparing the obtained results against those obtained by the εEDag [20], the most competitive approach in the aforementioned special session, which in fact uses a gradient-based mutation as a local search operator. In the second phase experiment, the Wilcoxon signed-rank test was computed to evaluate the confidence on the differences showed in both compared algorithms in all test problems.

The following parameters were used in MEMDEP: Population size $Pmax$ = 20, Cr = 0.9, F = 0.8, tolerance for equality constraints $\delta \in [1.0, 2.005e-5]$, cp = 46, dec = 1.003 and Gc = 1550. The dimensionalities used in the test problems were $D = 10$ and $D = 30$, as defined in the special session. The maximum number of evaluations was set to *Max_FES* = $200,000$ for 10D and $600,000$ for 30D. Those abovementioned MEMDEP parameters were defined by using the SPOT-R tool [32] on each test function and computing the average value for each single parameter. The initial values for each parameter required by the SPOT-R tool where those suggested in [20] and [30]. MEMDEP was coded in Java with Windows 7 OS and was run in a PC computer with an Intel Pentium Dual 2.19GHz processor.

The results for the first experiment are summarized in Tables I, II, and III for 10D, and in Tables IV, V, and VI for 30D. In both cases, the best, median, worst, average function values and standard deviation are shown when the fitness function evaluations (FES) are 2×10^4, 1×10^5, and 2×10^5 for 10D. Regarding 30D, the FES are 6×10^4, 3×10^5, and 6×10^5.

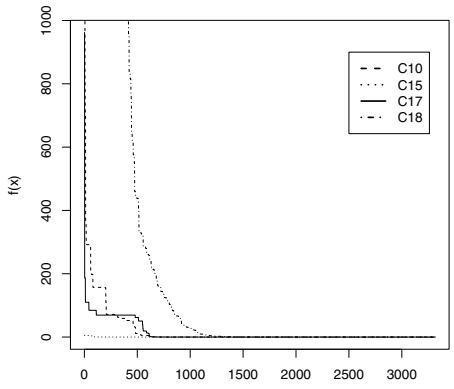

Fig. 1. Convergence plots of C10, C15, C17 and C18 test problems for 10D.

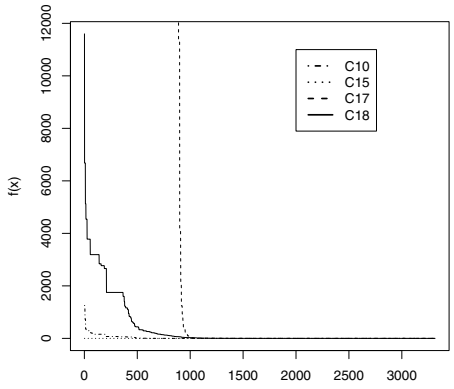

Fig. 2. Convergence plots of C10, C15, C17 and C18 test problems for 30D.

The aforementioned results suggest that MEMDEP is able to find feasible solutions with a short number of FES for both, 10D and 30D test problems. In twelve 10D test problems (C01, C02, C03, C07, C08, C09, C10, C13, C15, C16, C17, and C18) a fast approach to a very competitive feasible solution was observed (i.e., the best solution found at 20,000 FES is very similar to the final best solution after 200,00 FES and such solution is competitive with respect to that found by εEDag, as it will be discussed later in the paper). The exceptions were test problems C04 and C05, where feasible solutions were found after 20,000 FES. Moreover, in test problems C06, C11, C12, and C14, not all runs provided feasible solutions after 200,000 FES. However, it is worth noticing that for all eighteen test problems, at 100,000 FES, very competitive feasible solutions were found by MEMDEP, as it will be outlined in the next experiment.

The convergence plots of the run located in the median value of the 25 independent runs for four representative 10D test problems in Figure 1, show the best objective function value per iteration after the first feasible solution is found by MEMDEP. The same plots are presented in Figure 2 for 30D test problems. It can be noted that MEMDEP had a fast improvement in the feasible region once it was reached so as to locate competitive solutions before 1000 iterations.

TABLE I. FUNCTION VALUES ACHIEVED BY MEMDEP WHEN FES = 2×10^4, FES = 1×10^5, FES = 2×10^5 FOR 10D PROBLEMS C01-C06. "NA" MEANS NO FEASIBLE SOLUTIONS FOUND. (w) MEANS THAT IN ONLY w RUNS, OUT OF 25, FEASIBLE SOLUTIONS WERE FOUND.

FEs		C01	C02	C03	C04	C05	C06
$2.0E+04$	Best	-7.4731E-01	-2.1147E+00	7.6845E-03	NA	NA	-5.9081E+02 (5)
	Median	-7.4055E-01	-1.9070E+00	1.6143E-01	NA	NA	-5.9081E+02 (5)
	Worst	-7.0727E-01	-1.1966E+00	1.9445E+01	NA	NA	-5.9081E+02 (5)
	Average	-7.3951E-01	-1.8134E+00	1.7928E+00	NA	NA	-5.9081E+02 (5)
	Std	9.2840E-03	3.0058E-01	5.5607E+00	NA	NA	0.0000E+00 (5)
$1.0E+5$	Best	-7.4731E-01	-2.2780E+00	0.0000E+00	-4.7128E-04	-4.8357E+02	-5.7696E+02 (17)
	Median	-7.4056E-01	-2.2716E+00	0.0000E+00	-2.2644E-04	-4.7387E+02	-5.7167E+02 (17)
	Worst	-7.0728E-01	-1.1997E+00	4.0470E-25	9.1434E-01	-5.1808E+01	9.2543E+00 (17)
	Average	-7.3953E-01	-2.0828E+00	1.6863E-26	1.5420E-01	-3.5414E+02	-5.3498E+02 (17)
	Std	9.2769E-03	3.4562E-01	8.2610E-26	3.2448E-01	1.7212E+02	1.3156E+02 (17)
$2.0E+5$	Best	-7.4731E-01	-2.4172E+00	0.0000E+00	-4.7128E-04	-4.8359E+02	-5.7700E+02 (17)
	Median	-7.4056E-01	-2.2776E+00	0.0000E+00	-2.2644E-04	-4.2854E+02	-5.7192E+02 (17)
	Worst	-7.0728E-01	-1.1997E+00	4.0470E-25	9.1434E-01	5.4786E+01	9.2543E+00 (17)
	Average	-7.3954E-01	-2.1160E+00	1.6863E-26	1.5420E-01	-3.2522E+02	-5.3550E+02 (17)
	Std	9.2766E-03	3.4244E-01	8.2610E-26	3.2448E-01	1.9828E+02	1.3168E+02 (17)

TABLE II. FUNCTION VALUES ACHIEVED BY MEMDEP WHEN FES = 2×10^4, FES = 1×10^5, FES = 2×10^5 FOR 10D PROBLEMS C07-C12. "NA" MEANS NO FEASIBLE SOLUTIONS FOUND. (w) MEANS THAT IN ONLY w RUNS, OUT OF 25, FEASIBLE SOLUTIONS WERE FOUND.

FEs		C07	C08	C09	C10	C11	C12
$2.0E+04$	Best	3.8029E-03	1.6826E-05	1.2887E+12	4.9943E+11	-4.1453E+01 (6)	-3.7143E+03 (3)
	Median	1.2131E-01	1.0946E+01	2.4693E+12	1.8246E+12	-3.8019E+01 (6)	-3.5892E+03 (3)
	Worst	1.4804E+01	9.2936E+01	5.0840E+12	2.5594E+12	-3.3798E+01 (6)	-3.4641E+03 (3)
	Average	1.5922E+00	1.3171E+01	2.8278E+12	1.6278E+12	-3.7886E+01 (6)	-3.5892E+03 (3)
	Std	3.6789E+00	1.8845E+01	1.6171E+12	1.0440E+12	2.6002E+00 (6)	1.7688E+02 (3)
$1.0E+5$	Best	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	-3.3197E+01 (11)	-4.2668E+02 (6)
	Median	0.0000E+00	1.0573E+01	6.3109E-28	6.3235E-27	-7.1446E-03 (11)	-5.6623E+00 (6)
	Worst	3.9866E+00	9.2675E+01	5.0840E+12	5.2001E+12	-3.2911E-03 (11)	-2.7568E-01 (6)
	Average	4.7839E-01	1.1321E+01	4.8330E+11	4.8704E+11	-3.0241E+00 (11)	-1.2402E+02 (6)
	Std	1.3222E+00	1.8040E+01	1.2651E+12	1.2226E+12	1.0007E+01 (11)	1.9144E+02 (6)
$2.0E+5$	Best	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	-8.2053E-03 (20)	-4.2668E+02 (6)
	Median	0.0000E+00	1.0573E+01	6.3109E-28	6.3235E-27	-7.1372E-03 (20)	-5.6623E+00 (6)
	Worst	3.9866E+00	9.2675E+01	5.0840E+12	5.2001E+12	-3.2911E-03 (20)	-2.7568E-01 (6)
	Average	4.7839E-01	1.1321E+01	4.8330E+11	4.8704E+11	-6.8059E-03 (20)	-1.2402E+02 (6)
	Std	1.3222E+00	1.8040E+01	1.2651E+12	1.2226E+12	1.3640E-03 (20)	1.9144E+02 (6)

TABLE III. FUNCTION VALUES ACHIEVED BY MEMDEPWHEN FES = 2×10^4, FES = 1×10^5, FES = 2×10^5 FOR 10D PROBLEMS C13-C18. "NA" MEANS NO FEASIBLE SOLUTIONS FOUND. (w) MEANS THAT IN ONLY w RUNS, OUT OF 25, FEASIBLE SOLUTIONS WERE FOUND.

FEs		C13	C14	C15	C16	C17	C18
$2.0E+04$	Best	-6.8429E+01	9.0983E-02 (10)	5.8993E+00	1.0474E-10	1.0925E+00	3.1526E-06
	Median	-6.5578E+01	7.0788E+00 (10)	5.8993E+00	3.8551E-01	3.1902E+01	1.3105E+01
	Worst	-6.3517E+01	2.2144E+12 (10)	5.8993E+00	0.0000E+00	9.2503E+01	1.0066E+03
	Average	-6.5894E+01	1.4845E+11 (10)	5.8993E+00	4.4895E-01	3.6991E+01	1.7316E+02
	Std	2.4623E+00	5.7152E+11 (10)	0.0000E+00	5.2859E-01	3.2590E+01	2.8968E+02
$1.0E+5$	Best	-6.8429E+01	0.0000E+00 (20)	0.0000E+00	0.0000E+00	2.8350E-31	4.5251E-15
	Median	-6.5578E+01	0.0000E+00 (20)	0.0000E+00	0.0000E+00	1.0841E+00	7.4851E-09
	Worst	-6.3517E+01	2.0152E+12 (20)	6.5424E+13	1.0245E+00	7.9632E+01	1.0026E+03
	Average	-6.5894E+01	8.0613E+10 (20)	2.7519E+12	2.5649E-01	1.2333E+01	8.6854E+01
	Std	2.4623E+00	4.0303E+11 (20)	1.3350E+13	4.4411E-01	2.2421E+01	2.4217E+02
$2.0E+5$	Best	-6.8429E+01	0.0000E+00 (20)	0.0000E+00	0.0000E+00	0.0000E+00	4.5251E-15
	Median	-6.7004E+01	0.0000E+00 (20)	0.0000E+00	0.0000E+00	1.0841E+00	7.4851E-09
	Worst	-6.3518E+01	2.0152E+12 (20)	6.5424E+13	1.0245E+00	7.9632E+01	1.0026E+03
	Average	-6.6489E+01	8.0613E+10 (20)	2.7519E+12	2.5649E-01	1.2333E+01	8.6854E+01
	Std	2.3937E+00	4.0303E+11 (20)	1.3350E+13	4.4411E-01	2.2421E+01	2.4217E+02

TABLE IV. FUNCTION VALUES ACHIEVED BY MEMDEP WHEN FES = 6×10^4, FES = 3×10^5, FES = 6×10^5 FOR 30D PROBLEMS C01-C06. "NA" MEANS NO FEASIBLE SOLUTIONS FOUND. (w) MEANS THAT IN ONLY w RUNS, OUT OF 25, FEASIBLE SOLUTIONS WERE FOUND.

FEs		C01	C02	C03	C04	C05	C06
$6.0E+04$	Best	-8.1787E-01	-1.1012E+00	7.2131E+05 (4)	2.522E+00 (4)	5.573E+02 (2)	1.0054E+02 (2)
	Median	-7.7550E-01	-4.3551E-01	7.2131E+05 (4)	2.147E+01 (4)	5.613E+02 (2)	2.0462E+02 (2)
	Worst	-6.8921E-01	1.8762E+00	7.2131E+05 (4)	2.565E+01 (4)	5.721E+02 (2)	2.2326E+02 (2)
	Average	-7.6968E-01	-2.7742E-01	7.2131E+05 (4)	1.478E+01 (4)	5.630E+02 (2)	1.7614E+02 (2)
	Std	2.8686E-02	7.9886E-01	0.0000E+00 (4)	1.128E+01 (4)	7.093E+00 (2)	6.6129E+01 (2)
$3.0E+5$	Best	-8.1795E-01	-1.1012E+00	5.4360E+00 (4)	5.4608E-02 (4)	2.4864E+01 (2)	4.7135E+01 (2)
	Median	-7.7561E-01	-4.3551E-01	3.5512E+01 (4)	3.5122E-01 (4)	2.4864E+01 (2)	4.7138E+01 (2)
	Worst	-6.8926E-01	1.8762E+00	8.9172E+01 (4)	5.4615E-01 (4)	2.4864E+01 (2)	4.7140E+01 (2)
	Average	-7.6988E-01	-2.7742E-01	4.1408E+01 (4)	3.2580E-01 (4)	2.4864E+01 (2)	4.7138E+01 (2)
	Std	2.8720E-02	7.9886E-01	3.5156E+01 (4)	2.0969E-01 (4)	0.0000E+00 (2)	4.0945E-03 (2)
$6.0E+5$	Best	-8.1795E-01	-1.1012E+00	5.4360E+00 (4)	5.4608E-02 (4)	5.5735E+02 (2)	5.7707E+02 (2)
	Median	-7.7561E-01	-4.3551E-01	3.5512E+01 (4)	3.5122E-01 (4)	4.7334E+02 (2)	5.7707E+02 (2)
	Worst	-6.8926E-01	1.8762E+00	8.9172E+01 (4)	5.4615E-01 (4)	3.8933E+02 (2)	5.7707E+02 (2)
	Average	-7.6988E-01	-2.7742E-01	4.1408E+01 (4)	3.2580E-01 (4)	4.7334E+02 (2)	5.7707E+02 (2)
	Std	2.8721E-02	7.9886E-01	3.5156E+01 (4)	2.0969E-01 (4)	1.1880E+02 (2)	0.0000E+00 (2)

TABLE V. FUNCTION VALUES ACHIEVED BY MEMDEP WHEN FES = 6×10^4, FES = 3×10^5, FES = 6×10^5 FOR 30D PROBLEMS C07-C12. "NA" MEANS NO FEASIBLE SOLUTIONS FOUND. (w) MEANS THAT IN ONLY w RUNS, OUT OF 25, FEASIBLE SOLUTIONS WERE FOUND.

FEs		C07	C08	C09	C10	C11	C12
$6.0E+04$	Best	1.0488E+01	1.4407E+02	5.6760E+06 (15)	3.3756E+06 (15)	-1.6224E+00 (2)	5.9630E-02 (2)
	Median	7.0889E+01	2.7581E+03	1.6494E+13 (15)	2.4210E+12 (15)	-1.5897E+00 (2)	7.4598E-02 (2)
	Worst	8.1804E+02	1.0669E+05	3.4479E+13 (15)	3.6587E+13 (15)	-1.5570E+00 (2)	8.9567E-02 (2)
	Average	1.2024E+02	1.7616E+04	1.4011E+13 (15)	7.6062E+12 (15)	-1.5897E+00 (2)	7.4598E-02 (2)
	Std	1.8014E+02	2.7316E+04	1.2932E+13 (15)	1.1550E+13 (15)	4.6253E-02 (2)	2.1168E-02 (2)
$3.0E+5$	Best	3.7715E-12	4.7381E-07	5.6760E+06 (15)	3.3756E+06 (15)	-4.7643E-01 (2)	5.9630E-02 (2)
	Median	4.5681E-02	6.9881E+00	1.6494E+13 (15)	2.4210E+12 (15)	-1.0527E+00 (2)	2.1950E-01 (2)
	Worst	5.5741E+00	2.3110E+03	3.4479E+13 (15)	3.6587E+13 (15)	-1.6291E+00 (2)	3.7936E-01 (2)
	Average	1.3825E+00	1.9059E+02	1.4011E+13 (15)	7.6062E+12 (15)	-1.0527E+00 (2)	2.1950E-01 (2)
	Std	1.9884E+00	4.8023E+02	1.2932E+13 (15)	1.1550E+13 (15)	8.1503E-01 (2)	2.2609E-01 (2)
$6.0E+5$	Best	1.5518E-23	1.8189E-21	5.6760E+06 (15)	3.3756E+06 (15)	-4.7643E-01 (2)	5.9628E-02 (2)
	Median	4.2356E-15	3.9866E+00	1.6494E+13 (15)	2.4210E+12 (15)	-4.7643E-01 (2)	2.1949E-01 (2)
	Worst	3.9866E+00	8.2193E+02	3.4479E+13 (15)	3.6587E+13 (15)	-4.7643E-01 (2)	3.7936E-01 (2)
	Average	1.2757E+00	1.0611E+02	1.4011E+13 (15)	7.6062E+12 (15)	-4.7643E-01 (2)	2.1949E-01 (2)
	Std	1.8980E+00	1.8449E+02	1.2932E+13 (15)	1.1550E+13 (15)	0.0000E+00 (2)	2.2609E-01 (2)

TABLE VI. FUNCTION VALUES ACHIEVED BY MEMDEP WHEN FES = 6×10^4, FES = 3×10^5, FES = 6×10^5 FOR 30D PROBLEMS C13-C18.

FEs		C13	C14	C15	C16	C17	C18
$6.0E+04$	Best	-6.2194E+01	2.6656E+01	2.0512E+03	8.8662E-12	5.1657E-02	4.0352E+01
	Median	-5.4403E+01	2.5873E+13	1.1227E+14	5.4149E-10	6.3830E+02	5.6810E+03
	Worst	-3.4713E+01	9.3112E+13	5.1921E+14	1.0560E+00	1.1611E+03	9.2317E+03
	Average	-5.1042E+01	2.9017E+13	1.5592E+14	8.9564E-02	5.0081E+02	5.7079E+03
	Std	9.3764E+00	3.1201E+13	1.6690E+14	2.7526E-01	3.6022E+02	2.0731E+03
$3.0E+5$	Best	-6.3044E+01	1.9053E-08	2.1605E+01	0.0000E+00	3.8659E-02	3.7742E+01
	Median	-5.4443E+01	1.2271E+06	1.0246E+14	0.0000E+00	6.3830E+02	5.6810E+03
	Worst	-3.4809E+01	4.5671E+11	3.5672E+14	1.0560E+00	1.1611E+03	9.2317E+03
	Average	-5.1810E+01	4.9076E+10	1.1713E+14	8.9564E-02	5.0081E+02	5.6847E+03
	Std	9.0362E+00	1.2759E+11	1.2023E+14	2.7526E-01	3.6022E+02	2.0847E+03
$6.0E+5$	Best	-6.3044E+01	9.1089E-24	2.1603E+01	0.0000E+00	3.8659E-02	3.7742E+01
	Median	-5.4443E+01	3.9866E+00	1.0246E+14	0.0000E+00	6.3830E+02	5.6810E+03
	Worst	-3.4809E+01	1.0180E+09	3.5672E+14	1.0560E+00	1.1611E+03	9.2317E+03
	Average	-5.1810E+01	4.2952E+07	1.0400E+14	8.9564E-02	5.0081E+02	5.6847E+03
	Std	9.0362E+00	2.0321E+08	1.0677E+14	2.7526E-01	3.6022E+02	2.0847E+03

Regarding 30D test problems, in ten of them (C01, C02, C07, C08, C13, C14, C15, C16, C17, and C18), in a similar way as the behavior observed in 10D test problems, MEMDEP was able to quickly find feasible solutions. This was not the case in test problems C03, C04, C05, C06, C09, C10, C11, and C12, where feasible solutions were not consistently found. Nevertheless, even with some lack of robustness in some test problems, MEMDEP was able to generate a competitive feasible solution after 600,000 FES in all eighteen 30D test problems.

The results of the second experiment (comparison between MEMDEP and εEDag [20]) are presented in Tables VII and VIII, where the best results of both algorithms are compared at 20,000, 100,000, and 200,000 FES for 10D test problems, and at 60,000, 300,000, and 600,000 FES for 30D test problems. As it can be noted, MEMDEP was able to find better results with a lower number of FES for both, 10D and 30D test problems. Such behavior is evident at 100,000 FES in thirteen 10D test problems (CO2, C03, C04, C07, C08, C09, C10, C11, C14, C15, C16, C17, and C18) and at 60,000 FES in ten 30D test problems (C01, C07, C08, C12, C13, C14, C15, C16, C17, and C18).

The final results obtained by MEMDEP, after 200,000 FES for 10D and after 600,000 FEs for 30D test problems, were better with respect to those of εEDag in problems C02, C12 and C17 for 10D, and in problems C07, C08, C13, C14 and C17 for 30D. In the remaining problems either an equal or a very competitive result was found by the proposed MEMDEP. In fact, based on the results provided by the Wilcoxon Signed-Rank test using a 95%-confidence (showed at the bottom of Tables VII and VIII), the performance differences observed in both compared algorithms in all the 36 test problems are not significant, i.e., both of them are equally competitive in such benchmark. It is important to remark that MEMDEP uses a local search operator based on a direct-search method. In contrast, εEDag uses a gradient-based mutation which requires second-order information.

VII. Conclusions and Future Work

A Memetic DE-based algorithm for solving CNOPs has been introduced in this paper. An adaptation of the Powell's conjugate direction method (CDM) was used as a local search operator in three vectors from the current population (the best, the worst and one chosen at random) at each generation of the search. The modifications made to the CDM were a suitable stopping criteria to favor a low number of evaluations made by the local search operator, and the addition of the ε-constrained method within CMD as a constraint-handling technique. The proposed approach (MEMDEP) was used to solve eighteen benchmark problems with 10D and 30D (thirty six problems in total). Two experiments were carried out, one focused on analyzing the performance of the algorithm, and the other one centered on a comparison against εEDag, a highly competitive memetic algorithm also based on DE to solve CNOPs. The results suggest that MEMDEP is able to quickly reach the feasible region of the search space and also find a highly competitive feasible solution (even better than the one obtained by a competitive DE-based approach using a gradient-based local search operator). However, the main shortcoming of this first version of MEMDEP is a lack of robustness (i.e. for some test problems the feasible region was not consistently reached), and this is precisely the initial topic of future research, where other DE-variants and different ways to use CDM will be explored.

Acknowledgment

The authors acknowledge support from MinCyT bilateral project MX1103 and CONACyT bilateral project No. 164626.

References

[1] P. Siarry and Z. Michalewicz, Eds., *Advances in Metaheuristic Methods for Hard Optimization*. Berlin: Springer, 2008, iSBN 978-3-540-72959-4.

[2] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[3] J. Kennedy and R. C. Eberhart, *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[4] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, pp. 173–194, 2011.

[5] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivativess," *Computer Journal 7*, vol. 2, pp. 155–162, 1964.

[6] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[7] O. Kramer, "Iterated local search with powell´s method: a memetic algorithm for continuous global optimization," *Memetic Computing*, vol. 2, no. 1, pp. 69–83, 2010.

[8] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review." *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[9] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," 2002.

[10] A. E. Smith and D. W. Coit, "Constraint Handling Techniques—Penalty Functions," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford University Press and Institute of Physics Publishing, 1997, pp. C5.2–1–C5.2–6.

[11] S. Koziel and Z. Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[12] Z. Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints," in *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, D. B. Fogel, Ed. Piscataway, New Jersey: IEEE Press, 1995, pp. 647–651.

[13] M. Schoenauer and S. Xanthakis, "Constrained GA Optimization," in *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, S. Forrest, Ed., University of Illinois at Urbana-Champaign. San Mateo, California: Morgan Kauffman Publishers, July 1993, pp. 573–580.

[14] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2/4, pp. 311–338, 2000.

[15] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, September 2000.

[16] T. Takahama, S. Sakai, and N. Iwane, "Constrained optimization by the constrained hybrid algorithm of particle swarm optimization and genetic algorithm," in *AI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, S. Zhang and R. Jarvis, Eds. Springer Berlin Heidelberg, 2005, vol. 3809, pp. 389–400.

[17] T. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 3, pp. 284 –294, sep 2000.

[18] B. Liu, H. Ma, X. Zhang, and Y. Zhou, "A memetic co-evolutionary differential evolution algorithm for constrained optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, sept. 2007, pp. 2996 –3002.

[19] T. Takahama and S. Sakai, "Constrained optimization by the constrained differential evolution with gradient-based mutation and feasible elites," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 0-0 2006, pp. 1 –8.

[20] ——, "Constrained optimization by the epsilon constrained differential evolution with an archive and gradient-based mutation," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, july 2010, pp. 1 –9.

TABLE VII. BEST RESULT COMPARISON BETWEEN MEMDEP AND εDEAG [20] FOR 10D TEST PROBLEMS. "NA" MEANS NO FEASIBLE SOLUTION FOUND. BOLDFACE REMARKS THOSE BETTER RESULTS.

	$2.0E+04$		$1.0E+05$		$2.0E+05$	
Functions	εDeag [20]	MEMDEP	εDEag [20]	MEMDEP	εDeag [20]	MEMDEP
C01	-7.4705E-01	-7.4731E-01	-7.4731E-01	-7.4731E-01	-7.4731E-01	-7.4731E-01
C02	-2.0799E+00	**-2.1147E+00**	-2.2775E+00	**-2.2780E+00**	-2.2777E+00	**-2.2780E+00**
C03	2.5042E+01	**7.6845E-03**	4.4270E-16	**0.0000E+00**	0.0000E+00	0.0000E+00
C04	4.3228E+01	NA	3.1898E-03	**-4.7128E-04**	-9.9923E-06	-4.7128E-04
C05	-4.7867E+02	NA	-4.8361E+02	-4.8341E+02	-4.8361E+02	-4.8359E+02
C06	-5.1142E+02	**-5.9081E+02**	-5.7866E+02	-5.7696E+02	-5.7866E+02	-5.7700E+02
C07	4.8040E+00	**3.8029E-03**	4.3809E-18	**0.0000E+00**	0.0000E+00	0.0000E+00
C08	1.1645E+01	**1.6826E-05**	1.4689E-18	**0.0000E+00**	0.0000E+00	0.0000E+00
C09	1.0597E+02	1.2887E+12	6.1714E-18	**0.0000E+00**	0.0000E+00	0.0000E+00
C10	5.7125E+01	4.9943E+11	2.7211E-15	**0.0000E+00**	0.0000E+00	0.0000E+00
C11	3.0866E+00	-4.1453E+01	6.9866E-03	**-3.3197E+01**	-1.5227E-03	-8.2053E-03
C12	-7.9052E+02	-3.7143E+03	-8.8080E+02	-4.2668E+02	-5.7009E+02	**-4.2668E+02**
C13	-2.7444E+01	-6.8429E+01	-6.8429E+01	-6.8429E+01	-6.8429E+01	-6.8429E+01
C14	2.9291E+01	**9.0983E-02**	9.5225E-16	**0.0000E+00**	0.0000E+00	0.0000E+00
C15	9.8646E+01	**5.8993E+00**	6.1453E-15	**0.0000E+00**	0.0000E+00	0.0000E+00
C16	9.5086E-01	**1.0474E-10**	5.1753E-06	**0.0000E+00**	0.0000E+00	0.0000E+00
C17	1.5681E-01	1.0925E+00	3.6614E-05	**2.8350E-31**	1.4632E-17	**0.0000E+00**
C18	3.5543E-02	**3.1526E-06**	5.4517E-13	**4.5251E-15**	3.7314E-20	4.5251E-15
Wilcoxon Test Results	diff. not significant at $p \leq 0.05$.		diff. not significant at $p \leq 0.05$.		diff. not significant at $p \leq 0.05$.	

TABLE VIII. BEST RESULT COMPARISON BETWEEN MEMDEP AND εDEAG [20] FOR 30D TEST PROBLEMS. "NA" MEANS NO FEASIBLE SOLUTION FOUND. BOLDFACE REMARKS THOSE BETTER RESULTS.

	$6.0E+04$		$3.0E+05$		$6.0E+05$	
Functions	εDEag [20]	MEMDEP	εDEag [20]	MEMDEP	εDEag [20]	MEMDEP
C01	-7.3339E-01	**-8.1787E-01**	-8.1971E-01	-8.1795E-01	-8.2183E-01	-8.1795E-01
C02	-1.9941E+00	-1.1012E+00	-2.1679E+00	-1.1012E+00	-2.1692E+00	-1.1012E+00
C03	7.0330E+05	7.2130E+05	2.8744E+01	5.4360E+00	2.8673E+01	4.5535E+00
C04	2.4854E+01	2.5654E+01	1.8833E-01	5.4608E-02	4.6981E-03	4.3679E-02
C05	-4.0926E+02	5.5735E+02	-4.5274E+02	2.4864E+01	-4.5313E+02	5.5735E+02
C06	-4.5937E+02	1.0054E+02	-5.2846E+02	5.7707E+02	-5.2858E+02	5.7707E+02
C07	2.0785E+03	**1.0488E+01**	1.7062E+00	**3.7715E-12**	1.1471E-15	**1.5518E-23**
C08	5.2588E+04	**1.4407E+02**	5.1018E+00	**4.7381E-07**	2.5187E-14	**1.8189E-21**
C09	1.5139E+06	5.6760E+06	1.2472E-02	5.6760E+06	2.7707E-16	5.6760E+06
C10	1.1721E+06	3.3756E+06	3.2520E+01	3.3756E+06	3.2520E+01	3.3756E+06
C11	-1.8167E+00	-1.6224E+00	-2.4945E-02	-4.7643E-01	-3.2685E-04	-4.7643E-01
C12	7.0104E+02	**5.9630E-02**	-8.9577E+02	5.9630E-02	-1.9915E-01	5.9630E-02
C13	-6.5516E+00	**-6.2194E+01**	-5.9296E+01	**-6.2194E+01**	-6.6425E+01	-6.2194E+01
C14	5.2213E+06	**2.6656E+01**	6.2368E+00	**1.9053E-08**	5.0159E-14	**9.1089E-24**
C15	3.0062E+07	**2.0512E+03**	2.1617E+01	**2.1605E+01**	2.1603E+01	2.1603E+01
C16	2.4306E-02	**8.8662E-12**	2.7105E-19	**0.0000E+00**	0.0000E+00	0.0000E+00
C17	1.8741E+01	**5.1657E-02**	2.1657E-01	**3.8659E-02**	2.1657E-01	**3.8659E-02**
C18	7.1484E+01	**4.0352E+01**	1.2261E+00	3.7742E+01	1.2261E+00	3.7742E+01
Wilcoxon Test Results	diff. not significant at $p \leq 0.05$.		diff. not significant at $p \leq 0.05$.		diff. not significant at $p \leq 0.05$.	

[21] A. Mandal, A. Das, P. Mukherjee, S. Das, and P. Suganthan, "Modified differential evolution with local search algorithm for real world optimization," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, june 2011, pp. 1565 –1572.

[22] F. Solis and R. Wets, "Minimization by random search techniques," *Math. Oper. Res.*, vol. 6, no. 1, pp. 19–30, 1981.

[23] M. H. Fuqing Zhao and W. Ma, "A memetic differential evolution algorithm with adaptive mutation operator," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, no. 19, pp. 3687–3691, 2012.

[24] M. Pescador Rojas and C. A. Coello Coello, "A memetic algorithm with simplex crossover for solving constrained optimization problems," in *World Automation Congress (WAC), 2012*, june 2012, pp. 1 –6.

[25] Y. Z. Xiuqin Pan and X. Xu, "Adaptive differential evolution with local search for solving large-scale optimization problems," *Journal of Information and Computational Science*, vol. 9, no. 2, pp. 489–496, 2012.

[26] S. Hernández, G. Leguizamón, and E. Mezura-Montes, "Hibridación de evolución diferencial utilizando hill climbing para resolver problemas de optimización con restricciones," in *XVIII Argentine Congress on Computer Science*, Oct 2012, pp. 1–10, (in Spanish).

[27] S. Muelas, A. La Torre, and J.-M. Peña, "A memetic differential evolution algorithm for continuous optimization," in *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, ser. ISDA '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1080–1084.

[28] A. Menchaca-Méndez and C. A. Coello Coello, "A new proposal to hybridize the nelder-mead method to a differential evolution algorithm for constrained optimization," in *IEEE 2009 Congress on Evolutionary Computation (CEC'2009)*. Trondheim, Norway: IEEE Service Center, May 2009, pp. 2598–2605.

[29] K. Deb, *Optimization for Engineering Design Algorithms and Examples*, fisrt edition ed. PHI Learning PVT Ltd, 2005.

[30] E. Mezura-Montes, M. Damián-Araoz, and O. Cetina-Domínguez, "Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, july 2010, pp. 1–8.

[31] R. Mallipeddi and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2010.

[32] T. Bartz-Beielstein, "Spot: An r package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization," *CoRR*, vol. abs/1006.4645, 2010.